WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Thursday, January 04, 2007

Hide?	Set Name	Query	Hit Count
		DB=PGPB,USPT,USOC; PLUR=NO; OP=OR	
☐	L71	L70 and ((extent or extents) near page$)	2
☐	L70	(L66 or l67 or l68 or l69) and ((search$ or quer$ or request$ or inquir$ or enquir$ or question$) near (extent or extents))	88
☐	L69	707/104.1.ccls.	5788
☐	L68	707/100.ccls.	4986
☐	L67	707/5.ccls.	2185
☐	L66	707/1.ccls.	5398
☐	L65	(L63 or L64) and (search$ or quer$ or request$ or inquir$ or enquir$ or question)	89
☐	L64	L62 and ((extent or extents) near page$)	2
☐	L63	L62 and ((database$ or (data adj1 base$)) with table$)	90
☐	L62	(L60 or L61) and (buffer adj1 cach$)	250
☐	L61	(707/200 |707/201 |707/202 |707/203 |707/204 |707/205).ccls.	7771
☐	L60	(707/2 |707/3 |707/4).ccls.	10652
☐	L59	L58 and engine$	28
☐	L58	L57 and (search$ or quer$ or request$ or inquir$ or enquir$ or question)	28
☐	L57	L56 and (buffer adj1 cach$)	28
☐	L56	(database adj1 engine$)	2558
☐	L55	L54 and engine$	1
☐	L54	20030041214.pn.	1
☐	L53	L52 and (engine$ near (search$ or quer$ or request$ or inquir$ or enquir$ or question$))	36
☐	L52	(buffer adj1 cach$)	2391
☐	L51	L50 and (buffer near cach$)	22
☐	L50	L49 and (engine$ near (search$ or quer$ or request$ or inquir$ or enquir$ or question$))	4621
☐	L49	((database$ or (data adj1 base$)) with table$)	58059
☐	L48	L47 and ((database$ or (data adj1 base$)) with table$)	18
☐	L47	L46 and cach$	104
☐	L46	L45 and (engine$ near (search$ or quer$ or request$ or inquir$ or enquir$ or question$))	221
☐	L45	((extent or extents) with (page or pages))	3264

10/763,752

☐	L44	L43 and (extent or extents)	4
☐	L43	L41 and (buffer with cach$)	19
☐	L42	L41 and (buffer near cach$)	1
☐	L41	L40 and (engine$ near (search$ or quer$ or request$ or inquir$ or enquir$ or question$))	1608
☐	L40	(international adj1 business).asn.	66702
☐	L39	2002198872.pn.	0
☐	L38	2002198872.pn.	0
☐	L37	L36 and (engine$ near (search$ or quer$ or request$ or inquir$ or enquir$ or question$))	19
☐	L36	((extent or extents) near page$)	213
☐	L35	L26 and ((extent or extents) near page$)	2
☐	L34	L33 and page$	11
☐	L33	(L29 or L30) and ((database$ or (data adj1 base$)) with table$)	12
☐	L32	(L29 or L30) and (memory with (database$ or (data adj1 base$)) with table$)	0
☐	L31	L26 and (memory with (database$ or (data adj1 base$)) with table$)	2
☐	L30	L28 and (extent or extents).ab.	60
☐	L29	L28 and (extent or extents).ti.	1
☐	L28	(engine$ near (search$ or quer$ or request$ or inquir$ or enquir$ or question$))	21010
☐	L27	L26 and engine$	1
☐	L26	(buffer near cach$ near (extent or extents))	2
☐	L25	L15 and (search$ or request$ or inquir$ or enquir$ or question$ or quer$)	2
☐	L24	L22 and page$	1
☐	L23	(L18 or L19 or L20) and engine$	0
☐	L22	L21 and engine$	1
☐	L21	L11 and ((search$ or quer$ or request$ or inquir$ or enquir$ or question$) near (extent or extents))	5
☐	L20	L11 and ((extent or extents) same (buffer adj1 cach$))	2
☐	L19	L11 and ((extent or extents) with (buffer adj1 cach$))	2
☐	L18	L11 and ((extent or extents) near (buffer adj1 cach$))	1
☐	L17	L11 and ((search$ or quer$ or request$ or inquir$ or enquir$ or question$) with (extent or extents) with (buffer adj1 cach$))	0
☐	L16	L15 and engine$	1
☐	L15	L14 and (memory with (database$ or (data adj1 base$)) with table$)	2
☐	L14	L13 and (database$ or (data adj1 base$))	6
☐	L13	((buffer adj1 cache) with (extent or extents))	11
☐	L12	((buffer adj1 cache) near (extent or extents))	2

DB=USPT; PLUR=NO; OP=OR

(5317727 5812996 5822749 5758149 5794229 5794228 5655080 6374232

☐	L11	6973452 6101497 6442551 6021426 5903898 5956705 6457020 6470330 6243710 5668987 6073129 6105033 6122627 6134540 6226637 6226637 6285997 6341281 6470344 6477527 6574639 6801905 6889234 5832508 5737536 5826253 5850507 6182241 6898608 6957177 5201046 5511190 5742806 5918225 6289334 5884303 6654752 6961729 7010308 6389513 5787418 5842209).pn. (6078926 5426747 5940289 6125209 5581704 5615362 5706506 5802524 5832475 5941947 5944780 5974129 5999946 6009271 6070165 6122628 6128648 6134541 6185557 6279033 6282281 6360214 6381627 6401090 6411966 6438562 6449657 6466570 6487641 6532490 6539382 6601062 6604096 6694306 6694322 6732117 6741997 6763357 6898603 6950823 7062480 6728840 6912636 5799210 6230220 6591351 6760824 6904503 6347312 6748386).pn. (6014655 6115705 5210870 5237661 5454105 5530883 5537622 5537604 5537603 5548769 5590362 5619713 5745915 5758146 5778353 6457000 6691101 6754825 6836845 5418940 5802599 6049848 5907846 5010478 5283894 5542078 5781897 5918224 6115703 6928451 5561778 5579499 5590319 5594881 5600831 5701460 5737591 6134018 6253195 5506984 5694608 5893125 6138112 5317731 5495606 5546576 5560007 5596744 5634053 5664173).pn. (5666528 5724570 5412806 5603025 5692182 5692174 5727196 5787416 5845288 5870752 5894311 5903887 5937401 5950188 6006224 6012064 6044370 6076092 6081801 6212526 5201048 5265244 5329626 5367675 5423022 5504885 5574900 5596745 5615337 5619688 5627959 5632015 5694591 5701456 5701453 5749079 5761493 5761653 5774692 5794231 5797136 5799310 5806066 5826077 5835904 5842197 5842196 5852821 5878426 5897622).pn. (5905982 5918232 5924089 5930795 5930764 5937415 5943666 5953715 5956727 5960426 5966695 5974407 5974418 5987454 5991754 5995958 5995957 5995973 6006214 6009265 6009428 6012054 6016488 6023695 6023696 6026391 6044216 6047285 6047291 6078925 6081799 6085189 6088694 6092061 6105025 6108647 6108648 6112198 6125360 6134543 6134546 6138120 6148296 6192370 6199062 6199063 6212526 6249783 6249791 6263339).pn. (6272487 6282547 6298342 6324533 6353826 6356889 6363387 6381605 6421658 6427123 6430556 6453269 6460043 6470287 6470335 6477525 6477540 6496819 6502088 6507834 6516310 6549907 6557012 6560593 6571232 6581052 6581060 6584476 6598059 6615202 6615206 6618719 6631386 6633882 6636846 6691166 6697818 6708179 6708186 6714938 6732084 6735582 6735598 6741982 6748377 6757670 6799184 6801850 6807546).pn.	297
	DB=PGPB,USPT,USOC; PLUR=NO; OP=OR		
☐	L10	L9 and ((search$ or request$ or inquir$ or enquir$ or question$ or quer$) with (extent or extents))	10
☐	L9	l8 and (database$ or (data adj1 base$))	45
☐	L8	((table or tables) with (extent or extents) with (page or pages))	90
☐	L7	((database$ or (data adj1 base$)) with (table or tables) with (extent or extents) with (page or pages))	3
☐	L6	L3 and L5	11
☐	L5	((re-order$ or reorder$) near (page or pages))	143
☐	L4	((re-order$ or reorder$) near (extent or extents))	19
		((database$ or (data adj1 base$)) with (page or pages) with (quer$ or search$ or	

☐	L3	question$ or inquir$ or enquir$ or request$))	6438
☐	L2	((database$ or (data adj1 base$)) with (extent or extents) with (quer$ or search$ or question$ or inquir$ or enquir$ or request$) with (buffer adj1 cache))	1
☐	L1	((database$ or (data adj1 base$)) with (extent or extents) with (quer$ or search$ or question$ or inquir$ or enquir$ or request$))	325

END OF SEARCH HISTORY

PORTAL
USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: [✓ The ACM Digital Library] The Guide

extent and pages and table and database and query and buffe

SEARCH

Feedback Report a problem Satisfaction survey

Terms used
extent and **pages** and **table** and **database** and **query** and **buffer cache**

Found **51,668** of **193,448**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200   Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown   Relevance scale

1 Query evaluation techniques for large databases
Goetz Graefe
June 1993 **ACM Computing Surveys (CSUR)**, Volume 25 Issue 2
Publisher: ACM Press
Full text available: pdf(9.37 MB)   Additional Information: full citation, abstract, references, citings, index terms, review

Database management systems will continue to manage large data volumes. Thus, efficient algorithms for accessing and manipulating large sets and sequences will be required to provide acceptable performance. The advent of object-oriented and extensible database systems will not solve this problem. On the contrary, modern data models exacerbate the problem: In order to manipulate large sets of complex objects as efficiently as today's database systems manipulate simple records, query-processi ...

Keywords: complex query evaluation plans, dynamic query evaluation plans, extensible database systems, iterators, object-oriented database systems, operator model of parallelization, parallel algorithms, relational database systems, set-matching algorithms, sort-hash duality

2 I/O reference behavior of production database workloads and the TPC benchmarks—an analysis at the logical level
Windsor W. Hsu, Alan Jay Smith, Honesty C. Young
March 2001 **ACM Transactions on Database Systems (TODS)**, Volume 26 Issue 1
Publisher: ACM Press
Full text available: pdf(5.42 MB)   Additional Information: full citation, abstract, references, citings, index terms

As improvements in processor performance continue to far outpace improvements in storage performance, I/O is increasingly the bottleneck in computer systems, especially in large database systems that manage huge amoungs of data. The key to achieving good I/O performance is to thoroughly understand its characteristics. In this article we present a comprehensive analysis of the logical I/O reference behavior of the peak productiondatabase workloads from ten of the world's largest corporatio ...

Keywords: I/O, TPC benchmarks, caching, locality, prefetching, production database workloads, reference behavior, sequentiality, workload characterization

10\763,732

3 Interaction of query evaluation and buffer management for information retrieval
Björn T. Jónsson, Michael J. Franklin, Divesh Srivastava
June 1998 **ACM SIGMOD Record , Proceedings of the 1998 ACM SIGMOD international conference on Management of data SIGMOD '98**, Volume 27 Issue 2
Publisher: ACM Press
Full text available: pdf(1.81 MB) Additional Information: full citation, abstract, references, citings, index terms

The proliferation of the World Wide Web has brought information retrieval (IR) techniques to the forefront of search technology. To the average computer user, "searching" now means using IR-based systems for finding information on the WWW or in other document collections. IR query evaluation methods and workloads differ significantly from those found in database systems. In this paper, we focus on three such differences. First, due to the inherent fuzziness of the natural langua ...

4 B-tree concurrency control and recovery in page-server database systems
Ibrahim Jaluta, Seppo Sippu, Eljas Soisalon-Soininen
March 2006 **ACM Transactions on Database Systems (TODS)**, Volume 31 Issue 1
Publisher: ACM Press
Full text available: pdf(401.86 KB) Additional Information: full citation, abstract, references, index terms

We develop new algorithms for the management of transactions in a page-shipping client-server database system in which the physical database is organized as a sparse B-tree index. Our starvation-free fine-grained locking protocol combines adaptive callbacks with key-range locking and guarantees repeatable-read-level isolation (i.e., serializability) for transactions containing any number of record insertions, record deletions, and key-range scans. Partial and total rollbacks of client transactio ...

Keywords: ARIES, ARIES/CSA, B-tree, cache consistency, callback locking, client-server database system, data shipping, key-range locking, page server, partial rollback, physiological logging, sparse B-tree, structure modification

5 Data page layouts for relational databases on deep memory hierarchies
Anastassia Ailamaki, David J. DeWitt, Mark D. Hill
November 2002 **The VLDB Journal — The International Journal on Very Large Data Bases**, Volume 11 Issue 3
Publisher: Springer-Verlag New York, Inc.
Full text available: pdf(593.86 KB) Additional Information: full citation, abstract, index terms

Relational database systems have traditionally optimized for I/O performance and organized records sequentially on disk pages using the N-ary Storage Model (NSM) (a.k.a., slotted pages). Recent research, however, indicates that cache utilization and performance is becoming increasingly important on modern platforms. In this paper, we first demonstrate that in-page data placement is the key to high cache performance and that NSM exhibits low cache utilization on modern platforms. Next, we ...

Keywords: Cache-conscious database systems, Disk page layout, Relational data placement

6 GPGPU: general purpose computation on graphics hardware
David Luebke, Mark Harris, Jens Krüger, Tim Purcell, Naga Govindaraju, Ian Buck, Cliff Woolley, Aaron Lefohn
August 2004 **ACM SIGGRAPH 2004 Course Notes SIGGRAPH '04**

Publisher: ACM Press
Full text available: pdf(63.03 MB) Additional Information: full citation, abstract, citings

The graphics processor (GPU) on today's commodity video cards has evolved into an extremely powerful and flexible processor. The latest graphics architectures provide tremendous memory bandwidth and computational horsepower, with fully programmable vertex and pixel processing units that support vector operations up to full IEEE floating point precision. High level languages have emerged for graphics hardware, making this computational power accessible. Architecturally, GPUs are highly parallel s ...

7 Courses: State of the art in interactive ray tracing
Peter Shirley
July 2006 **Material presented at the ACM SIGGRAPH 2006 conference SIGGRAPH '06**
Publisher: ACM Press
Full text available: pdf(14.08 MB) Additional Information: full citation, abstract

Recent improvements in computer hardware have allowed ray tracing to be used in some interactive applications. The trends in architecture and expansions of geometric model should increase the use of interactive ray tracing. This course presents recent and often not-yet published work on interactive ray tracing.

8 Implementing sorting in database systems
Goetz Graefe
September 2006 **ACM Computing Surveys (CSUR)**, Volume 38 Issue 3
Publisher: ACM Press
Full text available: pdf(518.63 KB) Additional Information: full citation, abstract, references, index terms

Most commercial database systems do (or should) exploit many sorting techniques that are publicly known, but not readily available in the research literature. These techniques improve both sort performance on modern computer systems and the ability to adapt gracefully to resource fluctuations in multiuser operations. This survey collects many of these techniques for easy reference by students, researchers, and product developers. It covers in-memory sorting, disk-based external sorting, and cons ...

Keywords: Key normalization, asynchronous read-ahead, compression, dynamic memory resource allocation, forecasting, graceful degradation, index operations, key conditioning, nested iteration

9 An analysis of database workload performance on simultaneous multithreaded processors
Jack L. Lo, Luiz André Barroso, Susan J. Eggers, Kourosh Gharachorloo, Henry M. Levy, Sujay S. Parekh
April 1998 **ACM SIGARCH Computer Architecture News , Proceedings of the 25th annual international symposium on Computer architecture ISCA '98**, Volume 26 Issue 3
Publisher: IEEE Computer Society, ACM Press
Full text available: pdf(1.57 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

Simultaneous multithreading (SMT) is an architectural technique in which the processor issues multiple instructions from multiple threads each cycle. While SMT has been shown to be effective on scientific workloads, its performance on database systems is still an open question. In particular, database systems have poor cache performance, and the addition of multithreading has the potential to exacerbate cache conflicts.This paper examines database performance on SMT processors using traces of th ...

10 Performance enhancements to a relational database system
Michael Stonebraker, John Woodfill, Jeff Ranstrom, Marguerite Murphy, Marc Meyer, Eric Allman
June 1983 **ACM Transactions on Database Systems (TODS)**, Volume 8 Issue 2
Publisher: ACM Press
Full text available: pdf(1.33 MB) Additional Information: full citation, abstract, references, citings, index terms

In this paper we examine four performance enhancements to a database management system: dynamic compilation, microcoded routines, a special-purpose file system, and a special-purpose operating system. All were examined in the context of the INGRES database management system. Benchmark timings that are included suggest the attractiveness of dynamic compilation and a special-purpose file system. Microcode and a special-purpose operating system are analyzed and appear to be of more limited uti ...

Keywords: compiled query languages, database performance, file systems for databases, microcode

11 Computing curricula 2001
September 2001 **Journal on Educational Resources in Computing (JERIC)**
Publisher: ACM Press
Full text available: pdf(613.63 KB) html(2.78 KB) Additional Information: full citation, references, citings, index terms

12 Industrial session: potpourri: Getting priorities straight: improving Linux support for database I/O
Christoffer Hall, Philippe Bonnet
August 2005 **Proceedings of the 31st international conference on Very large data bases VLDB '05**
Publisher: VLDB Endowment
Full text available: pdf(349.39 KB) Additional Information: full citation, abstract, references, index terms

The Linux 2.6 kernel supports asynchronous I/O as a result of propositions from the database industry. This is a positive evolution but is it a panacea? In the context of the Badger project, a collaboration between MySQL AB and University of Copenhagen, we evaluate how MySQL/InnoDB can best take advantage of Linux asynchronous I/O and how Linux can help MySQL/InnoDB best take advantage of the underlying I/O bandwidth. This is a crucial problem for the increasing number of MySQL servers deployed ...

13 Accurate modeling of the hybrid hash join algorithm
Jignesh M. Patel, Michael J. Carey, Mary K. Vernon
May 1994 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 1994 ACM SIGMETRICS conference on Measurement and modeling of computer systems SIGMETRICS '94**, Volume 22 Issue 1
Publisher: ACM Press
Full text available: pdf(1.38 MB) Additional Information: full citation, abstract, references, citings, index terms

The join of two relations is an important operation in database systems. It occurs frequently in relational queries, and join performance is a significant factor in overall system performance. Cost models for join algorithms are used by query optimizers to choose efficient query execution strategies. This paper presents an efficient analytical model of an important join method, the hybrid hash join algorithm, that captures several key features of the algorithm's performance—including ...

14 Functional-join processing
R. Braumandl, J. Claussen, A. Kemper, D. Kossmann
February 2000 **The VLDB Journal — The International Journal on Very Large Data Bases**, Volume 8 Issue 3-4
Publisher: Springer-Verlag New York, Inc.
Full text available: pdf(486.22 KB) Additional Information: full citation, abstract, citings, index terms

Inter-object references are one of the key concepts of object-relational and object-oriented database systems. In this work, we investigate alternative techniques to implement inter-object references and make the best use of them in query processing, i.e., in evaluating functional joins. We will give a comprehensive overview and performance evaluation of all known techniques for simple (single-valued) as well as multi-valued functional joins. Furthermore, we will describe special *order-preser ...*

***Keywords:** Functional join, Logical OID, Object identifier, Order-preserving join, Physical OID, Pointer join, Query processing*

15 External memory algorithms and data structures: dealing with **massive data**
Jeffrey Scott Vitter
June 2001 **ACM Computing Surveys (CSUR)**, Volume 33 Issue 2
Publisher: ACM Press
Full text available: pdf(828.46 KB) Additional Information: full citation, abstract, references, citings, index terms

Data sets in large applications are often too massive to fit completely inside the computers internal memory. The resulting input/output communication (or I/O) between fast internal memory and slower external memory (such as disks) can be a major performance bottleneck. In this article we survey the state of the art in the design and analysis of external memory (or EM) algorithms and data structures, where the goal is to exploit locality in order to reduce the I/O costs. We consider a varie ...

Keywords: B-tree, I/O, batched, block, disk, dynamic, extendible hashing, external memory, hierarchical memory, multidimensional access methods, multilevel memory, online, out-of-core, secondary storage, sorting

16 The state of the art in distributed query processing
Donald Kossmann
December 2000 **ACM Computing Surveys (CSUR)**, Volume 32 Issue 4
Publisher: ACM Press
Full text available: pdf(455.39 KB) Additional Information: full citation, abstract, references, citings, index terms

Distributed data processing is becoming a reality. Businesses want to do it for many reasons, and they often must do it in order to stay competitive. While much of the infrastructure for distributed data processing is already there (e.g., modern network technology), a number of issues make distributed data processing still a complex undertaking: (1) distributed systems can become very large, involving thousands of heterogeneous sites including PCs and mainframe server machines; (2) the stat ...

Keywords: caching, client-server databases, database application systems, dissemination-based information systems, economic models for query processing, middleware, multitier architectures, query execution, query optimization, replication, wrappers

17 Join algorithm costs revisited
Evan P. Harris, Kotagiri Ramamohanarao
January 1996 **The VLDB Journal — The International Journal on Very Large Data Bases**, Volume 5 Issue 1
Publisher: Springer-Verlag New York, Inc.
Full text available: pdf(329.00 KB) Additional Information: full citation, abstract, citings, index terms

A method of analysing join algorithms based upon the time required to access, transfer and perform the relevant CPU-based operations on a disk page is proposed. The costs of variations of several of the standard join algorithms, including nested block, sort-merge, GRACE hash and hybrid hash, are presented. For a given total buffer size, the cost of these join algorithms depends on the parts of the buffer allocated for each purpose. For example, when joining two relations using the nested block j ...

Keywords: Join algorithms, Minimisation, Optimal buffer allocation

18 Distributed file systems: concepts and examples
Eliezer Levy, Abraham Silberschatz
December 1990 **ACM Computing Surveys (CSUR)**, Volume 22 Issue 4
Publisher: ACM Press
Full text available: pdf(5.33 MB) Additional Information: full citation, abstract, references, citings, index terms, review

The purpose of a distributed file system (DFS) is to allow users of physically distributed computers to share data and storage resources by using a common file system. A typical configuration for a DFS is a collection of workstations and mainframes connected by a local area network (LAN). A DFS is implemented as part of the operating system of each of the connected computers. This paper establishes a viewpoint that emphasizes the dispersed structure and decentralization of both data and con ...

19 Heuristic algorithms for I/O scheduling for efficient retrieval of large objects from tertiary storage
ChanHo Moon, Hyunchul Kang
January 2001 **Proceedings of the 12th Australasian database conference ADC '01**
Publisher: IEEE Computer Society
Full text available: pdf(760.58 KB) Publisher Site Additional Information: full citation, abstract, references, index terms

Multimedia data service applications of today are to efficiently deal with possibly massive amount of *large objects (LOBs).* The storage capacity of the traditional disk-based DBMS is certainly limited to support such applications. As such, it is necessary for the DBMS to employ the tertiary storage devices, which perform often with long latency and yet can provide huge amount of storage capacity at the relatively low cost. In this paper, we investigate the tertiary I/O scheduling algorith ...

20 Invited Tutorial 1: Context-sensitive program analysis as database queries
Monica S. Lam, John Whaley, V. Benjamin Livshits, Michael C. Martin, Dzintars Avots, Michael Carbin, Christopher Unkel
June 2005 **Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems**
Publisher: ACM Press
Full text available: pdf(183.53 KB) Additional Information: full citation, abstract, references

Program analysis has been increasingly used in software engineering tasks such as auditing programs for security vulnerabilities and finding errors in general. Such tools often require analyses much more sophisticated than those traditionally used in compiler optimizations. In particular, context-sensitive pointer alias information is a prerequisite for any sound and precise analysis that reasons about uses of heap objects in a program. Context-sensitive analysis is challenging becaus ...

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next